09/384,088 MO-892

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 :
G06F 15/415

A1

(11) International Publication Number: WO 92/15067

(43) International Publication Date: 3 September 1992 (03.09.92)

(21) International Application Number: PCT/GB92/00333

(22) International Filing Date: 24 February 1992 (24.02.92)

(30) Priority data:
9103937.0 26 February 1991 (26.02.91) GB

(71) Applicant (for all designated States except US): HEWLETT PACKARD COMPANY [US/US]; 3000 Hanover Street, Palo Alto, CA 94304 (US).

(72) Inventor; and
(75) Inventor/Applicant (for US only) : MARSHALL, David, Alan [GB/GB]; 5 Trin Mills, Merchants Landing, Bristol BS1 4RJ (GB).

(74) Agent: SQUIBBS, Robert, Francis; Hepworth Lawrence Bryer & Bizley, Lewins House, Lewins Mead, Bristol BS1 2NN (GB).

(81) Designated States: AT (European patent), BE (European patent), CH (European patent), DE (European patent), DK (European patent), ES (European patent), FR (European patent), GB (European patent), GR (European patent), IT (European patent), JP, LU (European patent), MC (European patent), NL (European patent), SE (European patent), US.

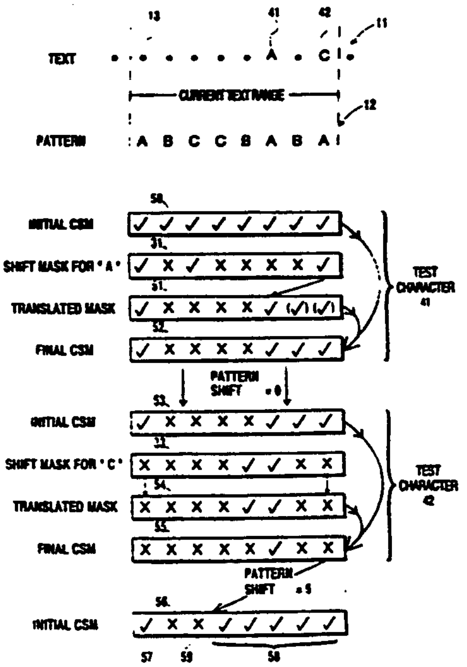
Published
With international search report.

Good Reference

(54) Title: SUBSTRING SEARCHING METHOD

(57) Abstract

A substring searching method is disclosed for locating within a character string (11), a substring that matches a given character pattern (12) of one or more characters, the characters making up the string and pattern being taken from the same character set. The method involves a preliminary phase in which a respective possible-match record (31) is formed for each character of the character set, this record indicating for each of the N last positions as the pattern is notionally moved up to the character associated with the record, to align the start of the pattern with the character, whether the character matches the corresponding character of the pattern. During a subsequent phase of the substring search method, the pattern (12) is notionally shifted along the string being searched (11) and at each position, a test is carried out one character at a time, to ascertain if the substring aligned with the pattern (12) matches the latter; whenever a character test indicates a mismatch, the pattern is shifted to a new position. The testing for a character match is effected by combining the possible match record (31) of a character of the substring being examined. With a current shift mask (50) that indicates the current state of knowledge concerning possible match positions, this combining operation involving forming an intermediate translated mask (51). The resultant new current shift mask (52) not only indicated whether a match is possible at the current position in view of the presence of the examined character, but also the amount by which the pattern should be shifted if a mismatch is indicated.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria
AU	Australia
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CS	Czechoslovakia
DE	Germany
DK	Denmark
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IT	Italy
JP	Japan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
LI	Liechtenstein
LK	Sri Lanka
LU	Luxembourg
MC	Monaco
MG	Madagascar
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
NL	Netherlands
NO	Norway
PL	Poland
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SN	Senegal
SU	Soviet Union
TD	Chad
TG	Togo
US	United States of America

SUBSTRING SEARCHING METHOD

Technical Field

The present invention relates to substring searching and, more particularly, to a method of locating within a character string a substring that matches a given character pattern of one or more characters, the characters making up the string and pattern being taken from the same character set.

The character set may, for example, be the English alphabet; in this case the given character pattern may be an English word it is desired to locate in a body of text constituting the character string. More generally, the character set could be all possible 256 combinations of 8 binary quantities (ie 8-bit bytes), substring searching being used to find occurrences of given byte patterns within a body of computer data. Alternatively, the character set could simply be the binary character set of 0,1 in which case substring searching becomes searching for a given binary sequence within a binary string.

Background Art

Substring searching has widespread uses in information processing ranging from pattern matching and feature recognition to text processing. As a result, there is a substantial body of prior art relevant to such searching.

The simplest algorithm for locating the occurrences of a pattern within a text will be called the "naive algorithm" here. Imagine the text as a line of data to be scanned linearly from left to right. The text has to be checked for a complete occurrence of the pattern at all possible positions. In the naive algorithm the pattern is first checked with its leftmost end aligned with the leftmost end of the text. Each character in the pattern, starting with the leftmost and proceeding through the pattern to the right one character at a time, is compared with the character in the corresponding position in the text. Whenever a position is found in which the pattern and text characters do not match, then the algorithm has determined that the pattern does not occur at this alignment in the text, and the pattern is moved one character to the right against the text. A new comparison of each pattern character with the corresponding text character is then begun, in order to check for the pattern at this new position in the text. If the end of the pattern is reached after all characters have been found to match then the algorithm has successfully located an occurrence of the pattern in the text. If the end of the text is reached without finding an occurrence of the pattern, then the algorithm has determined that the pattern does not occur in the text. This naive algorithm is conceptually simple but inefficient.

It will, of course, be appreciated that in the naive algorithm, searching could equally as well have been effected from right to left as from left to right. This is generally the case for substring searching methods and references herein to searching leftwards or rightwards should be taken as by way of example rather than by way of limitation, except insofar as these references imply a relative direction of execution of associated actions.

Knuth et al described a more efficient algorithm which will be called the "KMP" algorithm here (see: Knuth,D.E., Morris,J.H. and Pratt,V.R. Fast pattern matching in strings, Siam J Comput, Vol 6 No 2, June 1977, pp 323-350.)
The KMP algorithm differs from the naive algorithm in exploiting the fact that if in comparing the pattern with the text in a given position a number of characters are found to match before a mismatch is found, then the pattern of known text characters that have already been matched restricts the positions of pattern occurrences in the text that are possible. The KMP algorithm searches for the pattern in the text from left to right, and scans characters in the pattern from left to right one character at a time. As the scanning order is defined, the algorithm can, in a preliminary phase, precompute from the pattern a table of pattern-shift data that indicates for each position in the pattern at which a mismatch between the pattern and the text is first found the number of characters by which the pattern should be shifted to the right in order to find the

next possible occurrence of the pattern in the text. If the text is sufficiently long, the fixed overhead of precomputing the table is offset by the increased rate of testing possible positions for occurrence of the pattern in the main phase of the algorithm. The KMP algorithm is more efficient than the naive algorithm because it uses implicit information about the previously scanned characters to allow the pattern to be shifted more than one character at a time.

Boyer and Moore have also described an algorithm that uses the idea of precomputing tables of pattern-shift data in a preliminary phase of the algorithm, and that uses information about the text implicit in the number of characters already matched between the text and pattern in the current pattern position (see: Boyer,R.S. and Moore,J.S. A fast string searching algorithm. Commun ACM 20, 10 (Oct 1977), pp 762-772.). Boyer and Moore's algorithm, called the "BM" algorithm here, differed from the KMP algorithm in that the BM algorithm scans characters in the pattern from right to left, while still searching the text for pattern matches from left to right. The result of this simple difference is clearest if the first text character tested at a particular pattern position in the text happens to be a character that does not occur in the pattern at all. In the KMP algorithm the tested character is at the left hand end of the pattern, and the pattern is shifted one character to the right. In the BM algorithm the tested character is at the right hand end of the pattern, and the pattern is shifted to the right by the number of characters in the pattern. Any smaller shift of the pattern would place one of the characters in the pattern against the tested text character, which does not occur in the pattern.. In general, in the BM algorithm the pattern is shifted by using the text character that mismatched the pattern to index a precomputed table that gives the minimum pattern shift compatible with the value of the mismatched text character. Optionally a second precomputed table can be used to give the minimum pattern shift compatible with the fact that pattern characters previously tested in the current pattern position matched the text (and so these characters in the text must match any occurrence of the pattern in another position). If both precomputed tables are used then the pattern can be shifted by the maximum of the shifts indicated by the two tables. The BM algorithm is more efficient than the KMP algorithm because it uses a right to left scanning order, which on average results in the pattern being shifted by a larger number of characters on each character mismatch found.

Knuth et al in a postscript to his above-referenced article (on page 346) described a generalisation of the BM algorithm, which we will call the "BMKMP" algorithm. Knuth et al observed that when the BM algorithm shifts the pattern over the text, it "forgets" all that it already "knows" about the characters that have already been matched. They proposed the BMKMP algorithm which retains all of the knowledge of the text that is still relevant to the comparison process by encoding this knowledge into one of a finite number of states. Each state represents one of the possible combinations of known text characters at the current pattern position. The algorithm state determines the position of the next text character to be tested, and the value of that text character and the current state together determine the next state and by how many characters the pattern should be shifted against the text. This algorithm compares well with others in that it requires a low number of character comparisons between the pattern and the text. However, several factors make the BMKMP algorithm unattractive in practice :-

1. The state table is typically rather large. If all characters in the pattern are distinct, then there are (m^2 + m)/2 states, where there are m characters in the pattern and "^" denotes exponentiation. Worse, if all characters are not distinct then the only clear bound established on the size of the state table is 2^m states, which could be too large to implement in many cases.

2. Each state in the state table needs entries to define how the algorithm proceeds according to the character value found in the text. An entry is needed for each distinct character found in the pattern, and for all other characters together. The state table entries can be structured either to allow fast access to the entry for the character found in the text, or to allow efficient use of the memory used to store the state table. However, the aims of fast access and efficient use of memory are in

conflict to some extent, and this will make the BMKMP algorithm more expensive to implement than some of the other algorithms described here.

3. The state table must be computed from the pattern before the algorithm can begin searching the text. The size and complexity of the state table make this pre-computation an expensive operation, which could only be justified if a long text was to be searched.

Sunday has described a family of substring searching algorithms (called the "Sunday" algorithms here) which are also related to the BM algorithm (see: Sunday,D.M. "A very fast substring search algorithm" Commun ACM 33, 8 (Aug 1990), pp 132-142.). These algorithms differ from each other in the order in which the characters in the pattern are compared against the text, and these different comparison orders also result in different requirements for precomputing the tables of shift values. The BM algorithm uses one precomputed table to obtain a minimum value of pattern shift from the value of the text character that mismatched, and another to obtain a minimum value of pattern shift from the number of pattern characters successfully matched against the text at the current pattern position. In contrast, Sunday's algorithms use one precomputed table to obtain a minimum value of pattern shift from all of the pattern characters tested against the text at the current pattern position, including the mismatched character (although the value of text character found at the mismatch position is not used.) Once a mismatch is found, Sunday's algorithms read the text character immediately following the end of the pattern in its current position, and use this text character to access a second precomputed table which gives a second minimum value of pattern shift. As in the BM algorithm, the pattern is then shifted by the maximum of the two minimum pattern shift distances. These changes from the BM algorithm have the effect of allowing the characters in the pattern to be tested in any order, which allows Sunday to generate a family of substring searching algorithms as described. The different orders of testing the pattern characters result in different efficiencies. However, there are two aspects of all of Sunday's algorithms that result in some loss of theoretical efficiency :-

1. Like the BM algorithm, Sunday's algorithms "forget" everything they "know" about the text whenever they shift the pattern along the text. The forgotten information includes both the text characters compared with the pattern at its last position, and the value of the text character which immediately followed the end of the pattern in its last position and which was tested to determine the size of pattern shift.

2. Also like the BM algorithm, Sunday's algorithms shift the pattern by the larger of two separately determined minimum shift values.

It is an object of the present invention to provide a sub-string search method that overcomes at least certain of the drawbacks of the prior art methods.

Disclosure of the Invention

According to one aspect of the present invention, there is provided a method of locating within a character string a substring that matches a given character pattern of one or more characters, the characters making up the string and pattern being taken from the same character set, and said method comprising a preliminary phase during which pattern-shift data is derived from said given pattern, and a main phase in which the pattern is notionally positioned relative to said string at a succession of different current pattern positions in each of which one end of the pattern corresponds to a respective character position in said string and said pattern extends along the string therefrom in a predetermined direction, the main phase including the steps of :

(a) checking with the pattern in its current pattern position, for any mismatch between the pattern and the corresponding substring of said string, this step being first carried out with the pattern in an initial current pattern position in which said one end of the pattern corresponds to one end of the string, and

(b) where step (a) ascertains a mismatch, notionally shifting the pattern along the string in said predetermined direction, by an amount derived from said pattern-shift data, to a new current pattern position, steps (a) and (b) being repeated as necessary until a match is found or the string has been fully searched; **characterised in that:**

- said pattern-shift data is in the form of a respective possible-match record for each of a plurality of different character items of said character set where a character item may comprise a single character of said character set and/or combinations of such characters and said plurality of items is such as to enable any possible string to be built up from combinations, including repeats, of said character items,
- each possible-match record indicates, for an assumed situation of the associated said character item being present in the string and said one end of the pattern being moved in said predetermined direction towards the character item, whether said character item matches the corresponding character or characters of said pattern for each of the N last pattern positions taken by said pattern as its said one end is moved up to said character item, N being a positive integer, and
- the main phase involves determining, from the respective possible-match record of at least one character item of the substring that corresponds to said pattern in its current position, the next pattern position at which the presence of the, or all of the, said at least one character items is compatible with a match between the pattern and a substring of the string, this determination taking account of the position in said string of the or each said at least one character item relative to said one end of the pattern, by using an appropriate portion of the character item's possible-match record.

The possible-match records thus permit information on possible future matches derived from one or more character items to be utilised in determining the next pattern position. The use of possible-match records means that the order in which character items in the string are tested against the pattern can be varied. The possibility of combining possible-shift information derived in respect of several character items makes the method potentially powerful.

The said character items will often be constituted exclusively by all the single characters of the character set. However, in certain circumstances (for example, in searching strings made up from the binary character set of 0,1), it may be advantageous to use combinations of characters; thus for the binary character set case, the character items could be constituted by all possible 256 combinations of eight binary characters.

Generally, step (a) of the main phase involves identifying the character item present at a particular location in the substring currently corresponding to said pattern, and checking whether this character item matches the character or characters in the corresponding location in the pattern, these operations being repeated for different locations in said substring until either a mismatch is found or the whole substring has been successfully matched to the pattern. According to a preferred implementation of the invention, step (a) also includes providing a current possible-match indicator giving a cumulative indication of which pattern positions yet to be reached, provide the possibility of a match, this cumulative indication being provided by the incorporation into said indicator, for the or each character item identified, of the relevant portion of its possible-match record, said current possible-match indicator being used to determine said next pattern position to which the pattern position is shifted in step (b) of the main phase.

In this manner, a cumulative indication of the next possible match position is built up from possible-match information relevant to each character item tested in the current pattern position; in the event of a mismatch, the next pattern position to try is readily ascertained.

Advantageously, the said current possible-match indicator includes a possible match indication for the current pattern position as well as for pattern positions yet to be reached, this indication being cumulatively derived from the possible-match records of the or each character item identified for checking in step (a) of the main phase. The incorporation of this information into the current possible-match indicator, means that this indicator can not only be used to determine the next pattern position should a mismatch be found at the current position, but also to actually check for a mismatch at the current position in respect of each character item identified for checking. This leads to an efficient implementation of the present method.

Upon the pattern being shifted to a new current pattern position in step (b) of the main phase, the aforesaid said current possible-match indicator can either be reset (initially, to indicate all pattern positions not yet passed are possible match positions), or modified to retain only the portion thereof relevant to the pattern positions not yet passed.

At a low level, each said possible-match record and said current possible-match indicator preferably take the form of an N-bit field in which each bit position indicates whether a match is possible for a respective pattern position, the updating of said current possible-match indicator being effected by bit shifting and/or logically combining the relevant bits of a possible-match records with corresponding bits of said indicator.

Generally, the integer N will have a value equal to the character length of said pattern. However, it is also possible for N to have a value greater or less than the pattern length (the latter being required, for example, where the possible-match records are held in 32-bit memory locations and the pattern is more than 32 characters long).

Brief Description of the Drawings

A substring search method according to the present invention will now be particularly described, by way of non-limiting example, with reference to the accompanying diagrammatic drawings, in which:

Figure 1 is a diagram illustrating a predetermined pattern and a text string to be searched for occurrences of the pattern, the diagram being used to elucidate certain basic terms used in describing the substring search method;

Figure 2	is a diagram similar to Figure 1 but elucidating further terms used in describing the substring search method;
Figure 3	is a diagram illustrating the information content of shift masks used by the substring search method;
Figure 4	is a diagram illustrating the use of the shift masks to generate a cumulative current shift mask used by the substring search method;
Figure 5	is a top level flow diagram of the substring search method;
Figure 6	is a flow diagram of a shift-mask generation step of the Figure 5 flow diagram; and
Figure 7	is a flow diagram of a pattern shift calculation step of the Figure 5 flow diagram.

Best Mode for Carrying Out the Invention

In the substring search method now to be described, it is to be understood that the string (hereinafter referred to as the "text") to be searched for occurrences of the given pattern is held in a memory, which will be referred to as the "text memory". Referring to Figure 1, both the text 11 and pattern 12 are ordered sequences of characters, which we will describe as starting at the leftmost character and ending at the rightmost character (in the Figures, the characters of the text are generally represented by dots for clarity). The present substring search method scans the text from left to right, searching for the leftmost occurrence of the given pattern. If the pattern is not present in the text, this information is returned. If an occurrence of the pattern is found, then the search method can be reapplied to the text to find further occurrences of the given pattern if this is required. The substring search method tests for occurrence of the pattern with the leftmost character of the pattern aligned with the leftmost character of the text before any other alignment is tested. The position 13 of the text character that is currently aligned with the leftmost character of the pattern 12 during the operation of the search method will be described as the current "pattern position".

If the pattern contains M characters (in Figures 1 to 4, M=8), then at each pattern position the M characters of the pattern are aligned with an M-character substring of the text (called the "current text range" 14 here). If and only if each of the M pattern characters is equal in value to the text character with which it is aligned does the pattern position indicate an occurrence of the pattern in the text. If the method is to confirm the occurrence of the pattern at the current pattern position, then each of the pattern characters must be tested for equality with the text character with which it is aligned. In the present substring search method, the order (called the "pattern test sequence" here) in which the characters of the current text range are tested does not affect correct operation, and so this order can be chosen for maximum efficiency, and even varied throughout the operation of the method. A simple and reasonably effective choice of pattern test sequence is to test the characters in the current text range from rightmost to leftmost in sequence.

In the present substring search method, the comparisons between the text characters and the pattern characters are not made directly. Instead, the pattern is processed to provide a table of possible-match records each of which indicates for a respective text character whether the character matches with the corresponding pattern character for all possible relevant positions of pattern occurrences relative to that particular text character. This precomputed table and the text are then used to locate occurrences of the pattern in the text, the pattern itself not being required during the search. The table will be described as the "shift mask table" here and its possible-match records will be referred to as "shift masks". The shift mask table will now be described in more detail.

At any given pattern position, text characters that are to the left of the current text range have already been passed over by the pattern, and are no longer of interest. Text characters that are to the right of the current text range have not yet been reached by the pattern, and do not affect the operation of the search method at the current pattern position. Only text characters within the current text range will determine the operation of the search method at the current pattern position.

Conversely, a particular text character (that is, a character at a particular position within the text string) will only affect the operation of the search method at pattern positions for which that particular text character is aligned with a character of the pattern. This can be seen with reference to Figure 2 by considering the relationship between a particular text character 21 of the text 11 and the pattern 12 of M characters (note that the text character 21 has, for reasons of clarity, been represented in the text 11 as a "*", it being understood that the character may take on the character value of any of the characters making up the text).

The pattern 12 is illustrated in two pattern positions in Figure 2, these being a leftward pattern position, indicated by arrow 22L, in which the rightmost pattern character is in alignment with the text character 21, and a rightward pattern position, indicated by arrow 22R, which is the position of the text character 21. It will be readily appreciated that the pattern positions for which the text character will affect the operations of the search method are the M consecutive pattern positions starting with the pattern position indicated by arrow 22L and ending with the pattern position indicated by arrow 22R.

These M pattern positions will be described as the "mask range" 23 of the particular text character 21. For any given value of the text character 21, it can then be determined that the pattern can only occur in the text at those pattern positions within the mask range 23 of the text character 21 for which the pattern character aligned with the aforesaid particular text character 21 has the same character value as that text character. This knowledge is encapsulated in the shift mask table referred to above as will now be more fully explained with reference to Figure 3.

The shift mask table contains one shift mask for each character (that is, character value) in the set of characters from which the text can be composed. Each shift mask consists of M binary values called "mask bits". Each mask bit in a shift mask corresponds to one pattern position within the mask range of a text character, and indicates whether a pattern occurrence at that pattern position relative to the text character is consistent with the text character value being the same as the character value to which the shift mask relates. Figure 3 shows for each of three possible values of the test character 21 (namely, the values "A", "B" and "C"), the possible positions of pattern occurrence relative to occurrences of that character in the text; in addition, Figure 3 also shows the corresponding shift mask 31, 32, 33 for each of the three character values. In Figure 3 a tick within a shift mask indicates that the position of the tick corresponds to a possible position of a pattern occurrence, and a cross indicates that the corresponding pattern position is not a possible pattern occurrence. The rightmost mask bit in each shift mask corresponds to the rightmost pattern position within the mask range, in other words the pattern position for which the leftmost character of the pattern is aligned with the particular text character. The leftmost mask bit in each shift mask corresponds to the leftmost pattern position within the mask range, which is the pattern position for which the rightmost character of the pattern is aligned with the particular text character. The shift mask for any chosen character value can be computed by reversing the order of the pattern right to left, and then setting each mask bit of the shift mask to indicate whether the corresponding character in the reversed pattern is equal to the chosen character value.

Of course, for characters (character values) that are not present in the pattern, the corresponding shift masks each have all their mark bits set to indicate that a pattern occurrence is not possible.

The substring search method also maintains a "current shift mask", which records for the current pattern position which text characters positions within the current text range could be pattern positions corresponding to a pattern occurrence in the text. The current shift mask consists of M binary bits, each bit corresponding to one pattern position within the current text range. The leftmost mask bit in the current shift mask corresponds to the current pattern position, and the rightmost mask bit in the current shift mask corresponds to the pattern shifted (M-1) characters to the right of its current position, so that the leftmost character of the pattern would align with the same text character that the rightmost character of the pattern aligns with at the current pattern position. Positions of possible pattern occurrences outside the current text range do not need to be recorded. Possible pattern occurrences to the left of the current text range have already been checked, and at the current pattern position the search method has not tested any text characters that affect the possibility of pattern occurrences to the right of the current text range. All pattern occurrences to the right of the current text range are regarded as possible.

The current shift mask is the basic indicator of pattern matching possibilities during the execution of the search method, the current shift mask being updated for each new text character examined by combining elements of the corresponding character shift mask with the current shift mask. A general description of the operation of the search method will next be given followed by a specific example related to the pattern already illustrated in Figures 1 to 3.

Once the pattern and text strings have been read in and the shift mask table has been computed, the search method begins to test the text for occurrences of the pattern. The method tests for occurrence of the pattern with the leftmost character of the pattern aligned with the leftmost character of the text before any other alignment is tested. Nothing is yet known about the contents of the text, so the current shift mask is initialised to indicate that pattern occurrences are possible at all pattern positions.

The test for occurrence of the pattern at the first pattern position begins by reading the value of the text character that is aligned with the first pattern character in the pattern test sequence. The text character is used to retrieve the shift mask corresponding to that character from the shift mask table. This shift mask indicates the possible positions of pattern occurrences relative to the text character tested. Of course, the tested character may, as in the present case, be located in the text such that some of the pattern positions to which the corresponding shift mask relates are positioned to the left of the current pattern position and therefore of no relevance; however, since the tested character will be one lying within the current text range, there will always be a portion of the shift mask which is relevant to determining the possibility of pattern occurrences within the current text range.

The search method therefore now proceeds by taking the relevant portion of the shift mask of the tested character and combining it appropriately with the current shift mask so that indications in the two masks relating to the same pattern positions are brought together to generate a new current shift mask giving an updated indication on the possibility of pattern occurrences at the pattern positions embraced by the current shift mask.

The process of selecting the relevant portion of the shift mask to be combined with the current shift mask may be viewed in terms of initially aligning the two masks and then leftward shifting the shift mask of the tested character relative to the current shift mask until corresponding bits in the two masks refer to the possibility of pattern occurrence at the same pattern positions. The number of mask bits of shift required depend on the position of the text character read within the current text range. The bits of the shifted shift mask which align with bits of the current shift mask form the aforesaid relevant portion of the shift mask.

In practice, it is convenient to initially align the shift mask of the tested character with the current shift mask and then transform the shift mask into a new mask (herein, a translated mask) also aligned with the current shift mask, by shifting the shift mask an appropriate number of bits leftwards with bits shifted beyond the mask boundary being discarded and vacated positions at the righthand end of the mask all being set to indicate possible pattern occurrence. Thereafter, corresponding bits in the translated mask and current shift mask are logically combined to give a new value of the current shift mask, a pattern occurrence at each pattern position only being possible if it was possible before the most recently text character was read and if the translated mask indicates that the pattern occurrence at that pattern position is consistent with the newly known presence of the tested text character in the text. The logical combination could be implemented by bitwise ANDing or ORing of the two masks, depending on how the possibility of pattern occurrence is coded into binary bit values.

The new value of the current shift mask might or might not indicate that a pattern occurrence at the current pattern position is possible. If a pattern occurrence at the current pattern position is possible, then the last text character tested must have matched the pattern character with which it is aligned. In that case the search method proceeds to use the pattern test sequence to access further text characters for testing as described above. If all text characters in the current text range are tested and a pattern occurrence at the current pattern position is still indicated as being possible in the current shift mask, then a pattern occurrence at the current pattern position has indeed been found, and the current pattern position is returned. If the current shift mask indicates that a pattern occurrence at the current pattern position is not possible at any stage before pattern occurrence is confirmed, then the search method proceeds to test for pattern occurrence at the next possible pattern position. The next possible pattern position is indicated by the leftmost bit of the current shift mask that indicates possible pattern occurrence. The position of this leftmost bit within the current shift mask indicates the number of characters by which the pattern has to be shifted to the right along the text (called the "pattern shift" here).

Figure 4 illustrates the way in which the search method uses the current shift mask. In the situation shown in Figure 4, the pattern 12 is currently at position 13 in the text 11. Imagine that when the pattern first moved to this position the initial current shift mask (CSM) 50 indicated that all pattern positions in the current text range 14 were possible positions for pattern occurrences. Now let is be assumed that text character 41 is tested first, and is found to be an "A". The shift mask 31 for A (see Figure 3) is retrieved and has to be shifted two bits to the left to reflect the position of character 41 as being two characters to the left of the rightmost character in the current text range. The resultant translated mask 51 has its two rightmost bits (shown bracketed) set to indicate possible pattern occurrences at the corresponding pattern positions. The initial current shift mask 50 and the translated mask 51 are now combined to form a final current shift mask 52 for the character 41 test. The leftmost bit of mask 52 indicates that the current pattern position is a possible position of a pattern occurrence (because text character 41 matched the corresponding character in the pattern 12); in other words, the pattern shift is found to be zero. The search is therefore continued without shifting the pattern position, the next step being to test the next text character in the pattern test sequence - character 42 in this example. Because the pattern position is unchanged, the initial current shift mask 53 present at the start of character 42 testing, is the same as the final current shift mask 52 for character 41 testing. On testing, text character 42 is found to be a "C", and the shift mask 33 for C (see Figure 3) needs to be shifted by zero bits to form the corresponding translated mask 54 (this is because text character 42 is zero bits to the left of the rightmost character in the current text range - it is the rightmost character). The initial current shift mask 53 is then combined with the translated mask 54, marking pattern positions as possible positions of pattern occurrences only when both masks 33 and 43 indicated that the position was a possible pattern occurrence. The result is the new current shift mask 55 (the final CSM for character 42 testing).

This current shift mask 55 is then processed to find the value of the pattern shift which is five in this example. The current pattern position must now be shifted by five to the position of character 41 before testing is continued (it will be seen showing that this is indeed a possible position for a pattern occurrence, as both text characters 41 and 42 match the pattern at this new pattern position). The current shift mask must also be correspondingly shifted to form the initial current shift mask 56 for the next character test, the appropriate shift being five bits to the left to bring the leftmost bit indicating possible pattern occurrence (bit 57) to the leftmost bit position of the current shift mask 56. The bits 58 shifted into the right hand end of the current shift mask are all set to indicate that pattern occurrence at the corresponding pattern positions is possible, reflecting the fact that nothing is yet known about the corresponding parts of the text. This shifting procedure preserves all information about possible positions of pattern occurrence that is still of potential value - in this example the two bits 59 of the final current shift mask 56 indicate two pattern positions at which there cannot be occurrences of the pattern in the text. After the new current shift mask 56 is formed, the pattern test sequence is re-initialised, and testing continued at the new pattern position beginning by reading the value of the text character that is now aligned with the first pattern character in the pattern test sequence. The procedure is now as previously described, the search method terminating either on finding a pattern occurrence or on reaching the end of the text.

Figures 5, 6 and 7 are flow diagrams illustrating one possible implementation of the substring search method in program form. In the flow charts of Figures 5 to 7, the notation "A<<B" is used to indicate that element A has been shifted to the left by B bit positions with the rightmost bits of element A being set to indicate the possibility of pattern occurrence ("OK").

In the implementation illustrated in Figures 5 to 7, the pattern test sequence (that is, the order in which characters in the current text range at a particular pattern position, are tested) is determined by a one dimensional array PATTERN_TEST_SEQUENCE, the value of the nth element of which indicates the location in the current text range of the nth character to be tested. The location in the current text range corresponding to the current pattern position is assigned the value zero with successive locations having a successively incremented value.

Figure 5 is a top level flow chart of the substring search method, the first step 61 of this method being to read in the text and pattern. The next step 62 is to generate the shift mask table for all the characters making up the text string; this table is constituted by an array CHARMASK [] where CHARMASK [N] represents the shift mask for the character "N". The generation of the array CHARMASK [] is shown in more detail in Figure 6 to be described more fully hereinafter. The generation of the shift mask table is followed by a step 63 in which a variable CURRENT_SHIFT_MASK representing the current shift mask is initialized with all its bits set to represent possible pattern occurrence ("ALL OK"); at the same time a variable TEXT_POINTER representing pattern position is initialized to the start of the text string.

The search method then enters a double looped program structure, the outer loop of which (including steps 64 and 72) is concerned with advancing the search method along the text, and the inner loop of which (including steps 67 and 70) is concerned with testing for a match at the current pattern position. More particularly, step 64 ascertains whether there is text still left to be tested and if not, the result is returned at step 65 that no pattern has been found. However, if text is still left to be tested then a variable PATTERN_POINTER is initialized to one, this variable being used to indicate the number of the character next to be tested relative to how many characters have already been tested at the current pattern position. Thereafter, the inner loop concerning testing for a match at the current position is entered. The first step of the inner loop is to ascertain whether there are pattern positions still left to be tested (step 67); if this is not the case, then this will be because the whole pattern has been tested and found to match so that a result is returned at step 68 that a pattern has been found at the pattern position represented by the variable TEXT_POINTER. However, if there is still pattern left to test, then a next character within the current text range is selected (using the variable PATTERN_POINTER to access the pattern test sequence held in the array PATTERN_TEST_SEQUENCE), is tested and the resultant pattern shift determined, all this being done in step 69. The details of step 69 are shown in Figure 7, to be described more fully hereinafter.

If the determined pattern shift is zero (as tested at step 70) then this means that for the tested character, a pattern occurrence at the current pattern position is possible so that testing can continue at the current pattern position; to this end the variable PATTERN_POINTER is incremented to indicate the number of the next character to be tested (relative to how many characters have so far been tested at the current pattern position). After the variable PATTERN_POINTER has been incremented, step 67 is re-entered.

If the test carried out in program step 70 indicates that the pattern shift returned after a character test is non zero, then program step 72 is carried out to advance the pattern to a new pattern position by changing the pattern position pointer TEXT_POINTER and, at the same time, appropriately shifting the contents in the current shift mask. Following step 72, the search method returns to step 64.

Considering now the flow chart of Figure 6 illustrating the generation of the shift mask table, the first operation is to initialize all entries of the shift mask table array CHARMASK [CHAR] to a "NOT OK" setting, this being effected by steps 80 to 83 (the size of the array being determined by the number of characters in the alphabet from which the text string is composed). Following array initialisation, variables PATTERN_LOCATION and MASK_POSITION are also initialised, the variable PATTERN_LOCATION being set to reference the first pattern character position and the variable MASK_POSITION being set to the pattern length.

A loop is then entered to generate the shift masks on the basis of the characters in the pattern. The first step of this loop is to retrieve the pattern character at the location indicated by the value of the variable PATTERN_LOCATION (step 85). Next, the shift mask of the retrieved pattern character is modified by the setting of the bit at position MASK_POSITION within the mask to "OK" (step 86).

If the retrieved character was the last character of the pattern (step 87), generation of the shift mask table is now complete. However, if the end of the pattern has not been reached then the variable PATTERN_LOCATION is incremented and the variable MASK_POSITION is decremented (step 88) before control is returned to step 85.

Considering next the flow chart of Figure 7 which shows the details of the character-test and shift-calculation process represented by step 69 in Figure 5, the first step of this process is to set the value of a variable PATTERN_LOCATION to the value of the location of the character in the current text string next to be tested, this location value being determined by using the variable PATTERN_POINTER to access the array PATTERN_TEST_SEQUENCE (step 90). Next, the identity of the character at the text position offset from the current pattern position (indicated by TEXT_POINTER by the value held in PATTERN_LOCATION, is assigned to a variable TEXT_CHAR (step 91); this is the text character next to be tested against the pattern by use of the corresponding shift mask. To this end, the corresponding shift mask CHARMASK [TEXT_CHAR] is retrieved and the translated mask appropriate for the position of the character being tested is formed by shifting the retrieved shift mask to the left by a number of bits corresponding to the value (PATTERN_LENGTH - PATTERN_LOCATION) with the bits shifted into the righthand end of the mark being set to indicate the possibility of pattern occurrence.

Thereafter, in step 93 a new current shift mask is formed by combining the old current shift mask (CURRENT_SHIFT_MASK) with the translated mask TRANSLATED_MASK). Finally, in step 94, the pattern shift value SHIFT is determined by examining the newly formed current shift mask to ascertain the distance from the left of the mask of the first "OK" position. If there are no "OK" bits in the current shift mask, then SHIFT is set to the value of PATTERN_LENGTH.

It will be appreciated that the method of Figures 5 to 7 could be effected by a hardware equivalent to the illustrated program form.

The operation of the substring search method as described above is only illustrative of the present invention. Many variations of the method are possible. Some of these variants will now be described.

The pattern testing order, which is the order in which the text characters within the current text range are selected for reading, can be precomputed so as to obtain the maximum statistically expected pattern shift, considering the relative frequency of occurrence of the different pattern characters in normal usage of the alphabet. Optionally the relative positions of repeated characters within the pattern string can also be considered to obtain a more precise optimisation.

The pattern testing order could be dynamically optimised according to the pattern positions that are still possible positions for pattern occurrences at any stage. This optimisation may be too complex to be of practical value however.

The pattern testing order can be determined in any way which provides a suitable performance for the particular application of the searching method.

A record of text characters already tested could be maintained so as to avoid retesting them. A text character that has still to be tested according to the pattern testing sequence at the current pattern position might have been tested at a previous pattern position. If so, that text character must match the corresponding pattern character at the current pattern position, and so the text character does not have to be retested. The record of text characters already tested could take the form of a mask with a bit to indicate whether each character in the current text range had already been read. This mask would need to be shifted whenever the pattern and hence the current text range was shifted, and the appropriate bit of the mask would need to be set whenever a text character was read.

If the substring search method of the invention is implemented in computer software the various masks are most efficiently held in a single computer word each. This would imply that the masks have a limited length, say W bits (32 bits corresponding to patterns of up to 32 characters would be a common value for W using present computer technology.) Longer patterns might be handled either by using multiple computer words for each mask, or by maintaining masks that only represent the leftmost W characters at any pattern position and so only allow the pattern to be shifted by a maximum of W bits over the text between accesses to the text.

The pattern shift can be obtained from the current shift mask in various ways. If the search method of the invention was embodied in electronic hardware then dedicated circuitry could be used. If the search method was implemented in computer software then either a precomputed shift table can be used to relate the contents of the current shift mask (or parts of the contents of the current shift mask) to pattern shift values, or alternatively a binary interpolation method could be used to find the correct shift value. Both of these techniques and others are well known in the prior art, indeed, the instruction set of some computers includes an instruction that allows the position of the leftmost one bit in a word to be determined directly.

In certain circumstances (for example, in searching strings made up from the binary character set of 0,1), it may be advantageous to provide shift masks in respect of combinations of characters and then to check for matches in terms of these combinations; thus for the binary character set case, character combinations could used that constituted all possible 256 combinations of eight binary characters.

It will be appreciated that the substring searching method can be used where the given pattern includes a wildcard indicator indicating that the character at the corresponding position in the pattern may be any of a group of characters (which could be the whole character set or a specified subset of characters); in this case, the matching of character items with characters of said pattern will take account of which characters are included in said group. Furthermore, the method can also be used where the pattern includes a wildcard indicator indicating that the pattern includes an unspecified segment of one or more characters in length; in this case, the method of the invention would first be used to locate the pattern portion prefixing said segment and then implemented again to locate the pattern portion following the segment, this second search of the method being started from the end of the substring located by the first search.

CLAIMS

1. A method of locating within a character string a substring that matches a given character pattern of one or more characters, the characters making up the string and pattern being taken from the same character set, and said method comprising a preliminary phase during which pattern-shift data is derived from said given pattern, and a main phase in which the pattern is notionally positioned relative to said string at a succession of different current pattern positions in each of which one end of the pattern corresponds to a respective character position in said string and said pattern extends along the string therefrom in a predetermined direction, the main phase including the steps of :

(a) checking with the pattern in its current pattern position, for any mismatch between the pattern and the corresponding substring of said string, this step being first carried out with the pattern in an initial current pattern position in which said one end of the pattern corresponds to one end of the string, and

(b) where step (a) ascertains a mismatch, notionally shifting the pattern along the string in said predetermined direction, by an amount derived from said pattern-shift data, to a new current pattern position,

steps (a) and (b) being repeated as necessary until a match is found or the string has been fully searched; **characterised in that:**

- said pattern-shift data is in the form of a respective possible-match record for each of a plurality of different character items of said character set where a character item may comprise a single character of said character set and/or combinations of such characters and said plurality of items is such as to enable any possible string to be built up from combinations, including repeats, of said character items,

- each possible-match record indicates, for an assumed situation of the associated said character item being present in the string and said one end of the pattern being moved in said predetermined direction towards the character item, whether said character item matches the corresponding character or characters of said pattern for each of the N last pattern positions taken by said pattern as itş said one end is moved up to said character item, N being a positive integer, and

- the main phase involves determining, from the respective possible-match record of at least one character item of the substring that corresponds to said pattern in its current position, the next pattern position at which the presence of the, or all of the, said at least one character items is compatible with a match between the pattern and a substring of the string, this determination taking account of the position in said string of the or each said at least one character item relative to said one end of the pattern, by using an appropriate portion of the character item's possible-match record.

2. A method according to claim 1, wherein:

_ step (a) of the main phase involves identifying the character item present at a particular location in the substring currently corresponding to said pattern, and checking whether this character item matches the character or characters in the corresponding location in the pattern, these operations being repeated for different locations in said substring until either a mismatch is found or the whole substring has been successfully matched to the pattern; and

- step (a) also includes providing a current possible-match indicator giving a cumulative indication of which pattern positions yet to be reached, provide the possibility of a match, this cumulative indication being provided by the incorporation into said indicator, for the or each character item identified, of the relevant portion of its possible-match record, said current possible-match indicator being used to determine said next pattern position to which the pattern position is shifted in step (b) of the main phase.

3. A method according to claim 2, wherein:

- the said current possible-match indicator includes a possible match indication for the current pattern position as well as for pattern positions yet to be reached, this indication being cumulatively derived from the possible-match records of the or each character item identified in said substring in said step (a);

- said current possible-match indicator is updated, upon a said character item being identified, with the possible-match indications in the possible-match record of the character item, that relate to the current and yet to be reached pattern positions; and

- the said operation of checking whether an identified character item matches the character or characters in the corresponding location in the pattern, is effected by reference to the said current possible-match indicator.

4. A method according to claim 2 or 3, wherein upon the pattern being shifted to a new current pattern position in step (b) of the main phase, said current possible-match indicator is reset.

5. A method according to claim 2 or claim 3, wherein upon the pattern being shifted to a new current pattern position in step (b) of the main phase, said current possible-match indicator is modified to retain only the portion thereof relevant to the pattern positions not yet passed.

6. A method according to any one of claims 2 to 5, wherein each said possible-match record and said current possible-match indicator takes the form of an N-bit field in which each bit position indicates whether a match is possible for a respective pattern position, the updating of said current possible-match indicator being effected by bit shifting and/or logically combining the relevant bits of a possible-match records with corresponding bits of said indicator.

7. A method according to claim 1, wherein N has a value equal to the character length of said pattern.

8. A method according to claim 1, wherein said given pattern includes a wildcard indicator indicating that the character at the corresponding position in the pattern may be any of a group of characters, the matching of character items with characters of said pattern taking account of which characters are included in said group.

9. A method according to claim 1, wherein:

- step (a) of the main phase involves identifying the character item present at a particular location in the substring currently corresponding to said pattern, and checking whether this character item matches the character or characters in the corresponding location in the pattern, these operations being repeated for different locations in said substring until either a mismatch is found or the whole substring has been successfully matched to the pattern; and

- the said operation of checking whether an identified character item matches the character or characters in the corresponding location in the pattern, is effected by reference to the possible-match record of the character item and, in particular, to an indication contained in the record that relates to the current pattern position.

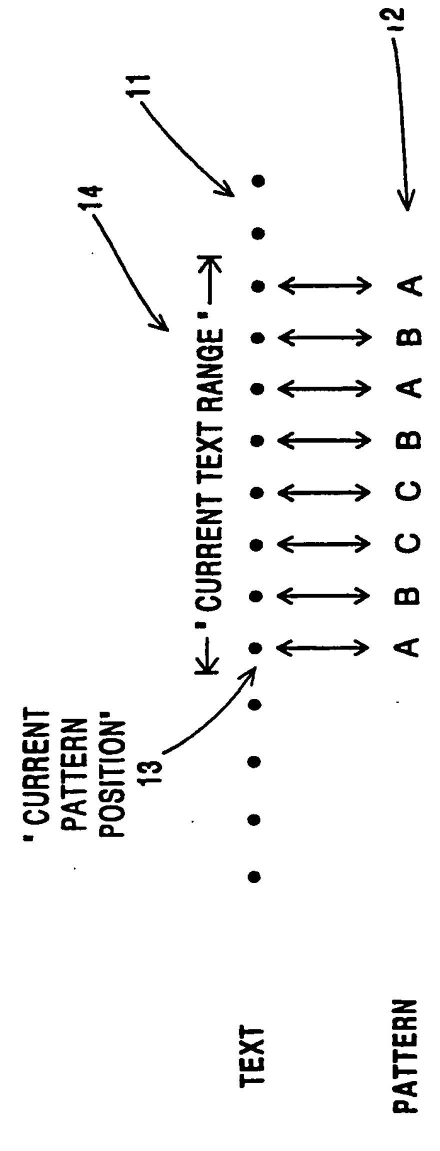

TEXT
PATTERN
'CURRENT PATTERN POSITION'
13
' CURRENT TEXT RANGE '
14
11
12
A
B
C
C
B
A
B
A

FIG 1

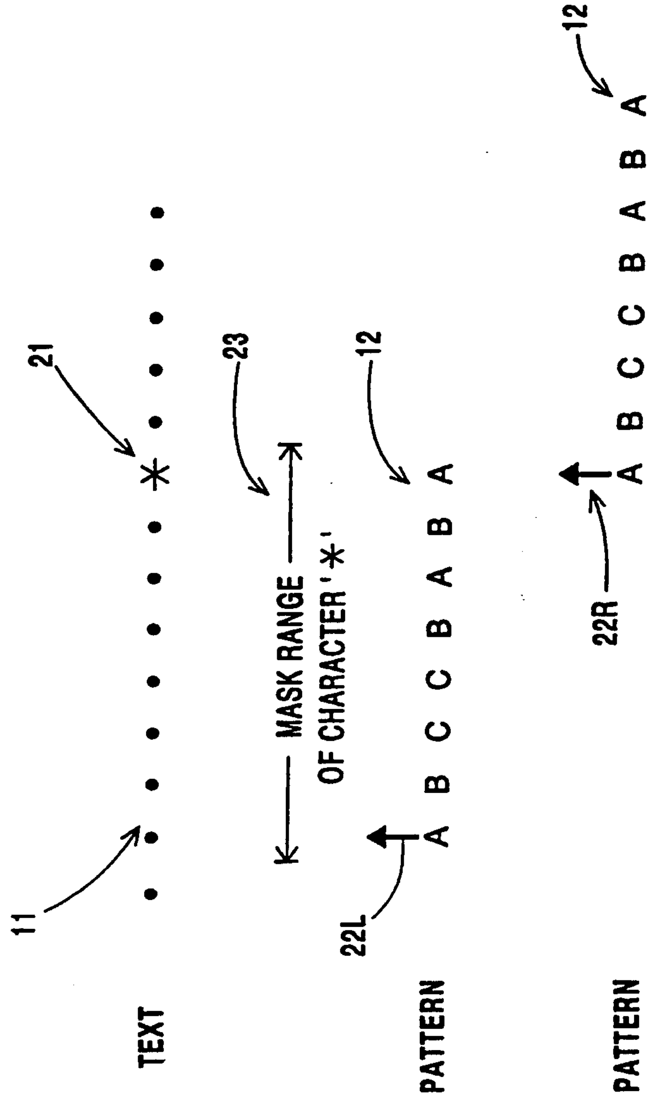
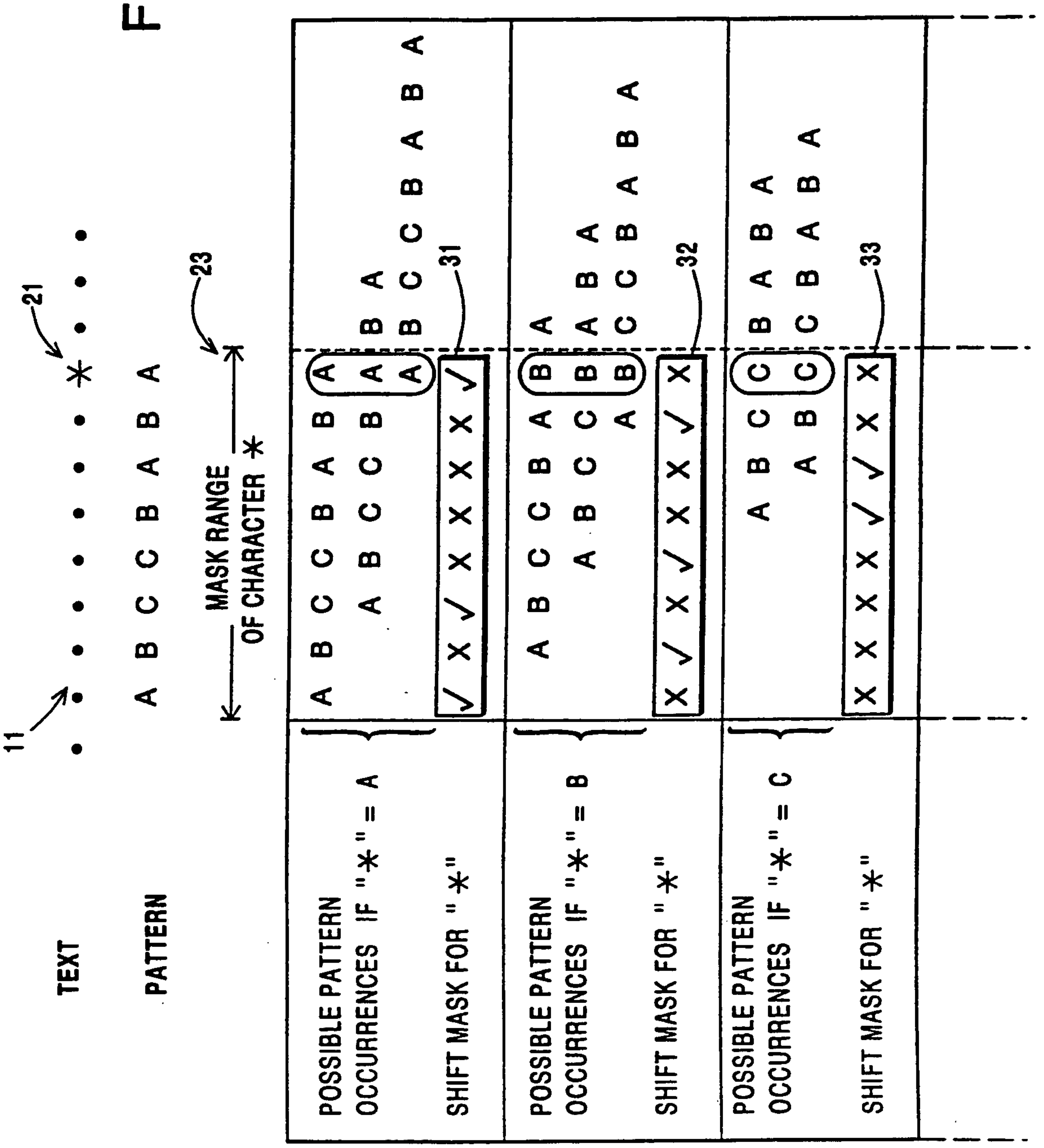

TEXT
11
21
*
MASK RANGE
OF CHARACTER '*'
23
PATTERN
22L
A B C C B A B A
12
PATTERN
22R
A B C C B A B A
12

FIG 2

FIG 3


TEXT
• • • • • • • • * • • •
11
21
PATTERN
A B C C B A B A
MASK RANGE OF CHARACTER *
23
POSSIBLE PATTERN OCCURRENCES IF "*" = A
A B C C B A B A
A B C C B A B A
A B C C B A B A
SHIFT MASK FOR "*"
✓ X ✓ X X X X ✓
31
POSSIBLE PATTERN OCCURRENCES IF "*" = B
A B C C B A B A
A B C C B A B A
A B C C B A B A
SHIFT MASK FOR "*"
X ✓ X ✓ X X ✓ X
32
POSSIBLE PATTERN OCCURRENCES IF "*" = C
A B C C B A B A
A B C C B A B A
SHIFT MASK FOR "*"
X X X X ✓ ✓ X X
33

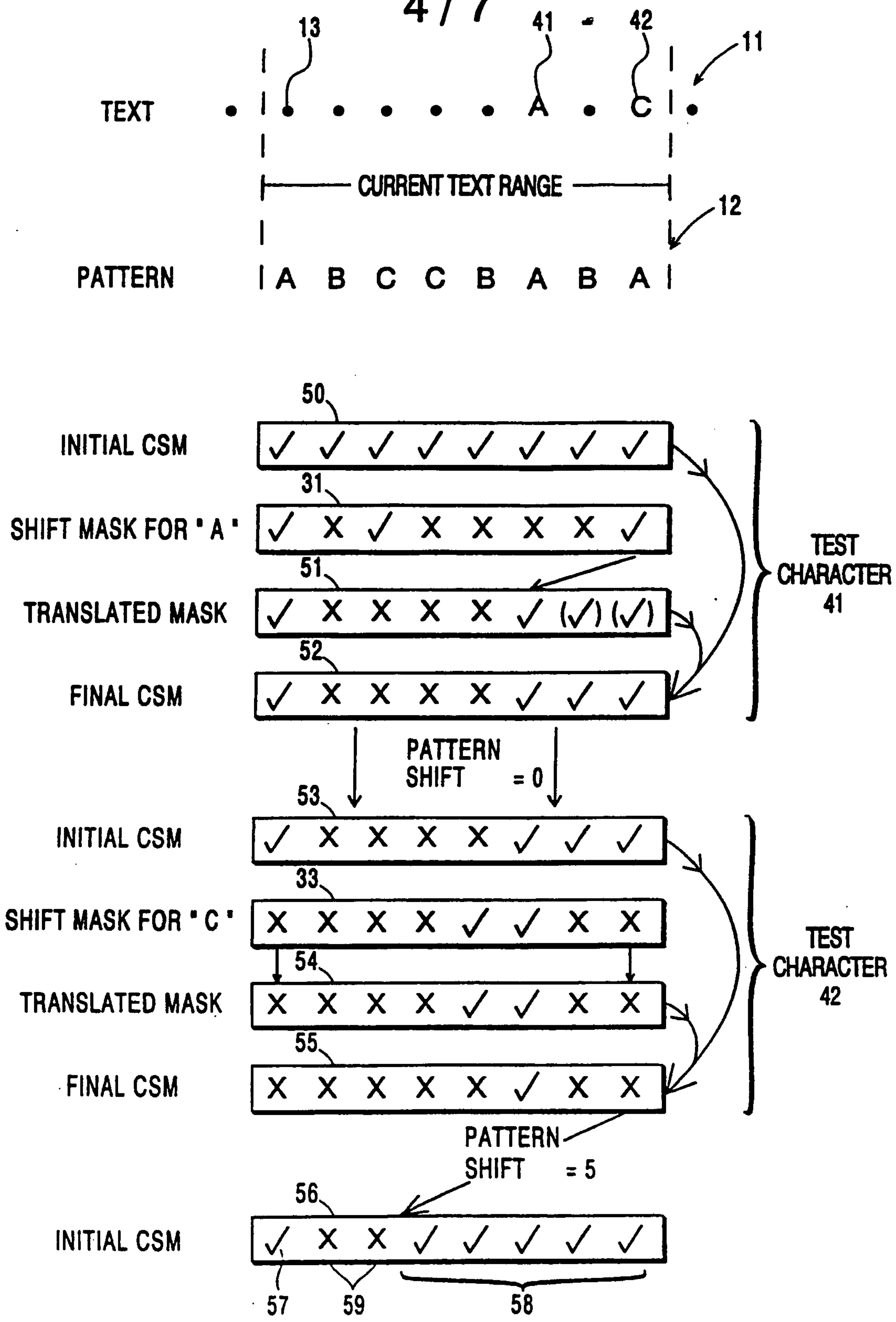

13
41
42
11
TEXT
• • • • • • A • C •
CURRENT TEXT RANGE
12
PATTERN
A B C C B A B A
50
INITIAL CSM
✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓
31
SHIFT MASK FOR " A "
✓ X ✓ X X X X ✓
51
TRANSLATED MASK
✓ X X X X ✓ (✓) (✓)
52
FINAL CSM
✓ X X X X ✓ ✓ ✓
TEST CHARACTER 41
PATTERN SHIFT = 0
53
INITIAL CSM
✓ X X X X ✓ ✓ ✓
33
SHIFT MASK FOR " C "
X X X X ✓ ✓ X X
54
TRANSLATED MASK
X X X X ✓ ✓ X X
55
FINAL CSM
X X X X X ✓ X X
TEST CHARACTER 42
PATTERN SHIFT = 5
56
INITIAL CSM
✓ X X ✓ ✓ ✓ ✓ ✓
57
59
58

FIG 4

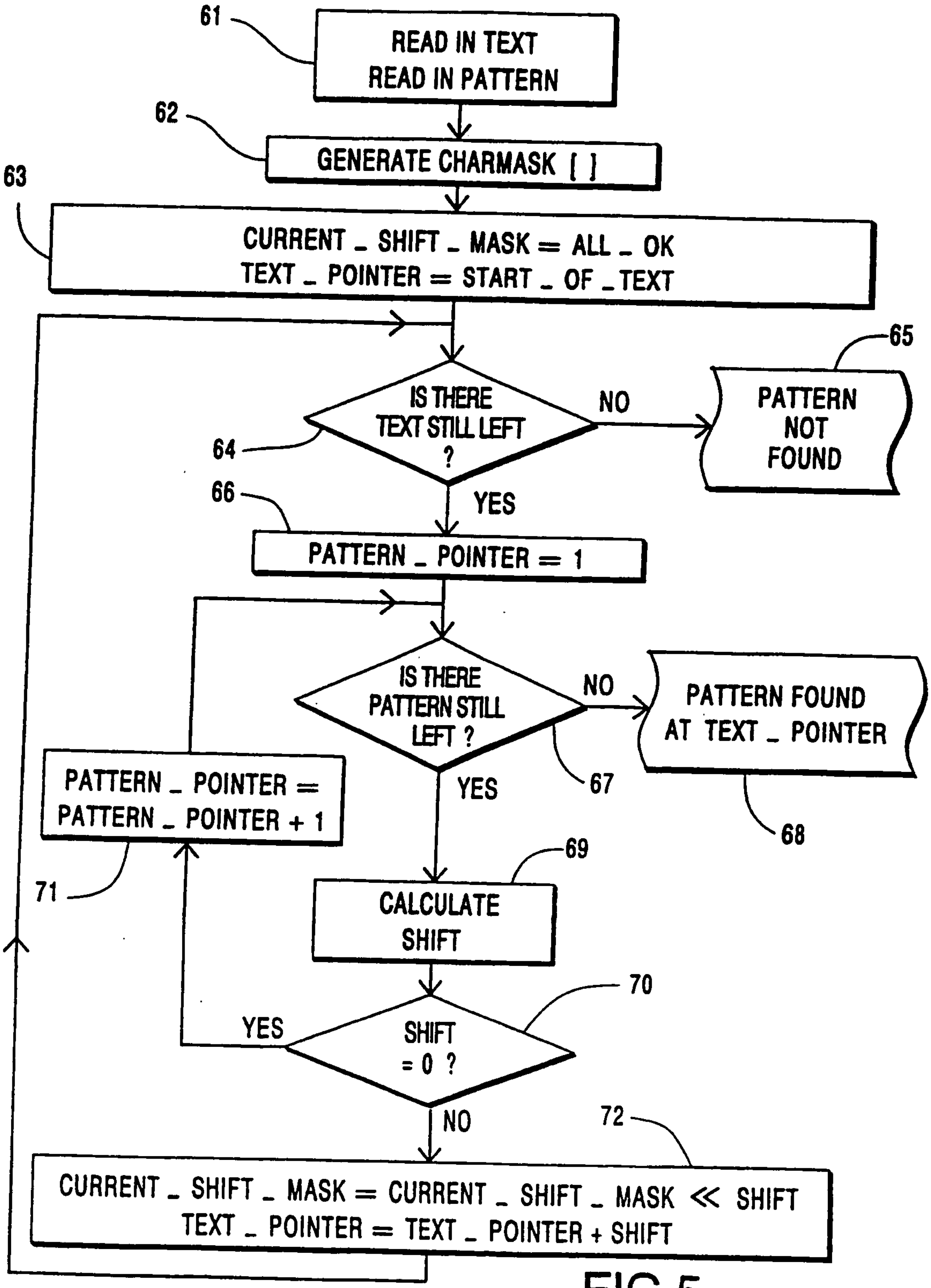

61
READ IN TEXT
READ IN PATTERN
62
GENERATE CHARMASK [ ]
63
CURRENT _ SHIFT _ MASK = ALL _ OK
TEXT _ POINTER = START _ OF _TEXT
64
IS THERE TEXT STILL LEFT ?
NO
65
PATTERN NOT FOUND
YES
66
PATTERN _ POINTER = 1
IS THERE PATTERN STILL LEFT ?
NO
PATTERN FOUND AT TEXT _ POINTER
67
68
YES
PATTERN _ POINTER = PATTERN _ POINTER + 1
71
69
CALCULATE SHIFT
70
YES
SHIFT = 0 ?
NO
72
CURRENT _ SHIFT _ MASK = CURRENT _ SHIFT _ MASK ≪ SHIFT
TEXT _ POINTER = TEXT _ POINTER + SHIFT

FIG 5

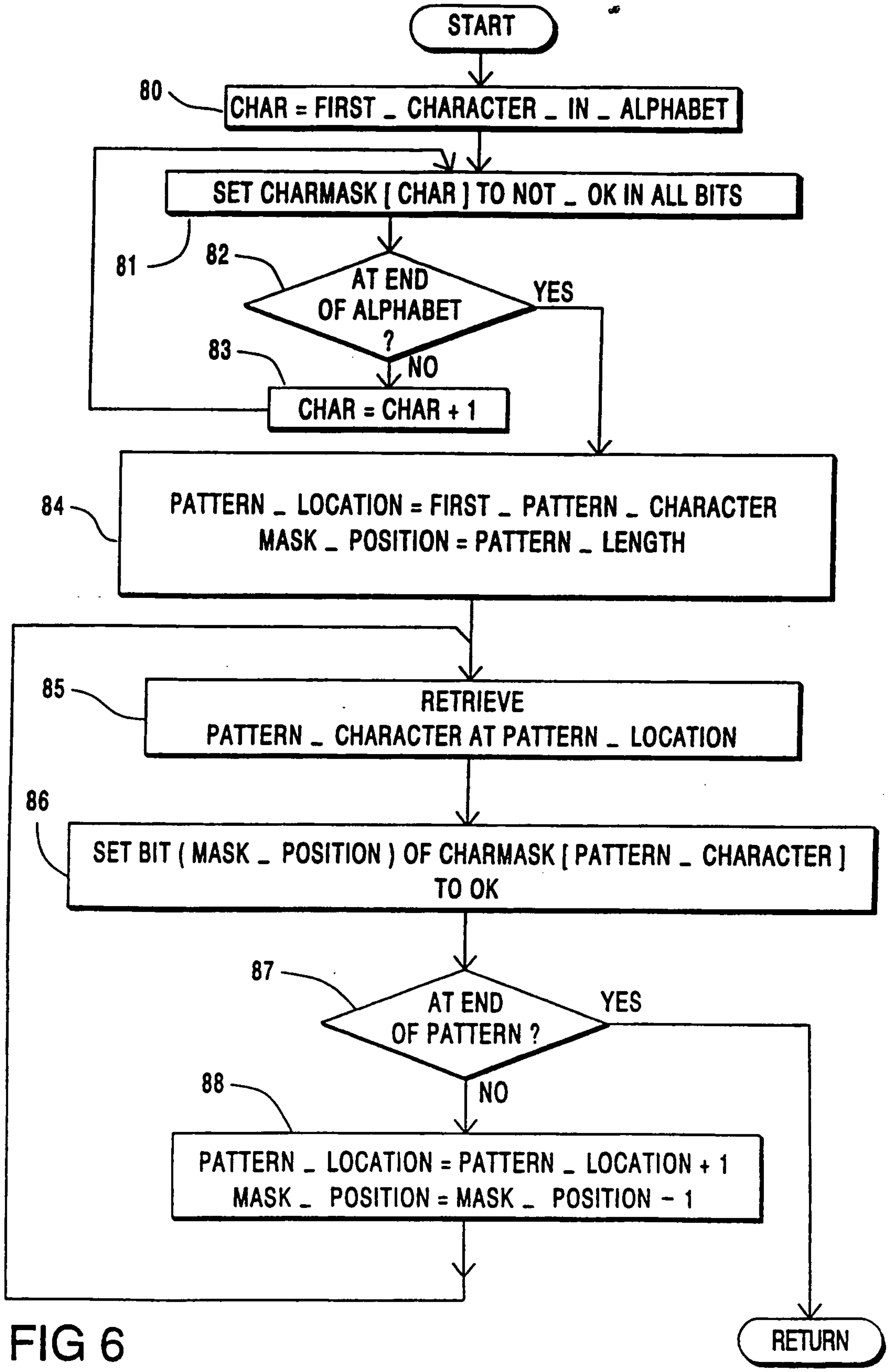

START
80
CHAR = FIRST _ CHARACTER _ IN _ ALPHABET
SET CHARMASK [CHAR] TO NOT _ OK IN ALL BITS
81
82
AT END OF ALPHABET ?
YES
NO
83
CHAR = CHAR + 1
84
PATTERN _ LOCATION = FIRST _ PATTERN _ CHARACTER
MASK _ POSITION = PATTERN _ LENGTH
85
RETRIEVE PATTERN _ CHARACTER AT PATTERN _ LOCATION
86
SET BIT (MASK _ POSITION) OF CHARMASK [PATTERN _ CHARACTER] TO OK
87
AT END OF PATTERN ?
YES
NO
88
PATTERN _ LOCATION = PATTERN _ LOCATION + 1
MASK _ POSITION = MASK _ POSITION - 1
RETURN

FIG 6

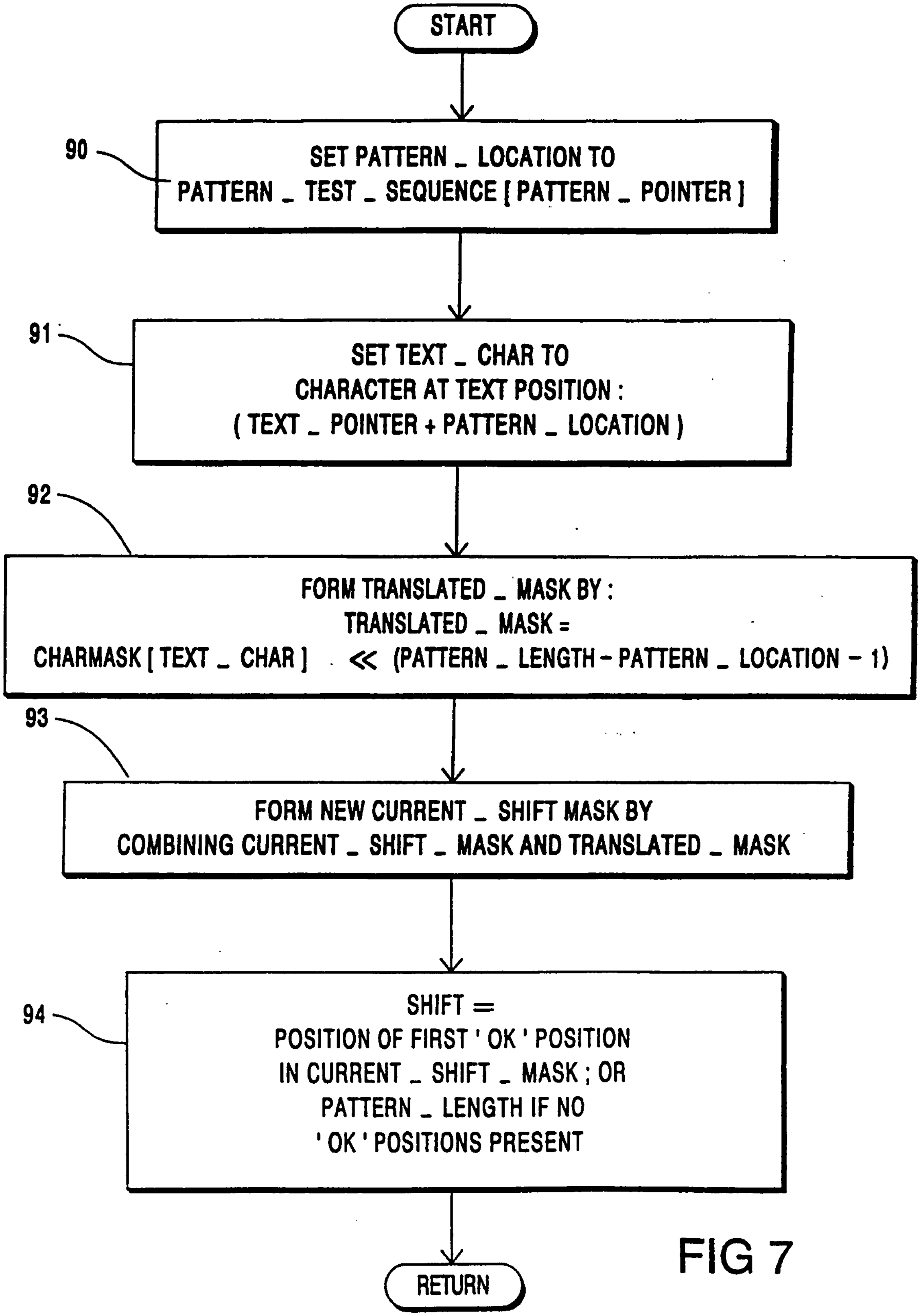

START
90
SET PATTERN _ LOCATION TO
PATTERN _ TEST _ SEQUENCE [PATTERN _ POINTER]
91
SET TEXT _ CHAR TO
CHARACTER AT TEXT POSITION :
(TEXT _ POINTER + PATTERN _ LOCATION)
92
FORM TRANSLATED _ MASK BY :
TRANSLATED _ MASK =
CHARMASK [TEXT _ CHAR] « (PATTERN _ LENGTH - PATTERN _ LOCATION - 1)
93
FORM NEW CURRENT _ SHIFT MASK BY
COMBINING CURRENT _ SHIFT _ MASK AND TRANSLATED _ MASK
94
SHIFT =
POSITION OF FIRST ' OK ' POSITION
IN CURRENT _ SHIFT _ MASK ; OR
PATTERN _ LENGTH IF NO
' OK ' POSITIONS PRESENT
RETURN

FIG 7

INTERNATIONAL SEARCH REPORT

International Application No PCT/GB 92/00333

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all)[6]

According to International Patent Classification (IPC) or to both National Classification and IPC

Int.Cl. 5 G06F15/415

II. FIELDS SEARCHED

Minimum Documentation Searched[7]

Classification System	Classification Symbols
Int.Cl. 5	G06F

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched[8]

III. DOCUMENTS CONSIDERED TO BE RELEVANT[9]

Category °	Citation of Document,[11] with indication, where appropriate, of the relevant passages[12]	Relevant to Claim No.[13]
A	SIAM J COMPUT vol. 6, no. 2, June 1977, pages 323 - 350; D.E. KNUTH ET AL.: 'Fast pattern matching in strings' cited in the application see the whole document ---	1
A	COMMUNICATIONS OF THE ACM vol. 20, no. 10, October 1977, pages 762 - 772; R.S. BOYER ET AL.: 'A fast string searching algorithm' cited in the application see the whole document --- -/--	1

° Special categories of cited documents :[10]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search 11 MAY 1992	Date of Mailing of this International Search Report 16 06 92
International Searching Authority EUROPEAN PATENT OFFICE	Signature of Authorized Officer KATERBAU R.E.

2

International Application No PCT/GB 92/00333

III. DOCUMENTS CONSIDERED TO BE RELEVANT (CONTINUED FROM THE SECOND SHEET)

Category °	Citation of Document, with indication, where appropriate, of the relevant passages	Relevant to Claim No.
A	COMMUNICATIONS OF THE ACM vol. 33, no. 8, August 1990, pages 132 - 142; D.M. SUNDAY: 'A very fast substring search algorithm' cited in the application see the whole document —	1